New-School Machine Structures

Big Idea: Memory Hierarchy

- Parallel Requests
  - Assigned to computer e.g., Search "Katz"
- Parallel Threads
  - Assigned to core e.g., Lookup, Ads
- Parallel Instructions
  - >1 instruction @ one time e.g., 5 pipelined instructions
- Parallel Data
  - >1 data item @ one time e.g., Add of 4 pairs of words
- Hardware descriptions
  - All gates @ one time

Smart Phone
Warehouse Scale Computer
Virtual Memory
Core
Memory (Cache)
Input/Output
Computer
Flank
Instruction Unit(s)
Functional Unit(s)
Main Memory
Logic Gates

Protection + Indirection = Virtual Address Space

Agenda

- Virtual Memory Revisted
- Administrivia
- Paging and Performance
- Exceptions, Traps, Interrupts
- Break
- Virtual Machines
- Summary

Protection + Indirection = Virtual Address Space

Agenda

- Virtual Memory Revisted
- Administrivia
- Paging and Performance
- Exceptions, Traps, Interrupts
- Break
- Virtual Machines
- Summary

Protection + Indirection = Virtual Address Space

Agenda

- Virtual Memory Revisted
- Administrivia
- Paging and Performance
- Exceptions, Traps, Interrupts
- Break
- Virtual Machines
- Summary

Protection + Indirection = Virtual Address Space
Protection + Indirection = Virtual Address Space

Protection + Indirection = Virtual Address Space

Protection + Indirection = Dynamic Memory Allocation

Protection + Indirection = Dynamic Memory Allocation

Protection + Indirection = Controlled Sharing

Protection + Indirection = Controlled Sharing

Protection + Indirection = Controlled Sharing

Protection + Indirection = Controlled Sharing

Protection + Indirection = Controlled Sharing
Day in the Life of an (Instruction) Address

No Cache, No Virtual Memory
(Note: PA - Physical Address, VA - Virtual Address)

Day in the Life of an (Instruction) Address

No Cache, Virtual Memory, TLB Hit—Very Fast!

Day in the Life of an (Instruction) Address

No Cache, Virtual Memory, TLB Miss, Page Table Access
NOTE: Translation before caching

Day in the Life of an (Instruction) Address

Physical Data Cache, Virtual Memory, TLB Miss, Page Table Access

VA caches are possible, but it's a little more complicated

Day in the Life of an (Instruction) Address

Physical Data Cache, Virtual Memory, TLB Miss, Page Table Access

VA caches are possible, but it's a little more complicated
Day in the Life of an (Instruction) Address

![Diagram of data access process]

Day in the life of a data access is not too different
Physical Data & Instruction Cache, Virtual Memory, TLB Miss, Page Table Access

What if there's a page fault?

Peer Instruction: True or False
A program tries to load a word at X that causes a TLB miss but not a page fault. Which are True or False:
1. A TLB miss means that the page table does not contain a valid mapping for virtual page corresponding to the address X
2. There is no need to look up in the page table because there is no page fault
3. The word that the program is trying to load is present in physical memory.

Red) 1 F, 2 F, 3 F, 4 T
Blue) 1 F, 2 F, 3 T, 4 F
Green) 1 F, 2 T, 3 F, 4 T
Yellow) 1 F, 2 T, 3 T, 4 T

Peer Instruction: True or False
TLBs entries have valid bits and dirty bits. Data caches have them also.
A. The valid bit means the same in both: if valid = 0, it must miss in both TLBs and Caches.
B. The valid bit has different meanings. For caches, it means this entry is valid if the address requested matches the tag. For TLBs, it determines whether there is a page fault (valid=0) or not (valid=1).
C. The dirty bit means the same in both: the data in this block in the TLB or Cache has been changed.
D. The dirty bit has different meanings. For caches, it means the data block has been changed. For TLBs, it means that the page corresponding to this TLB entry has been changed.

Red) 1 F, 2 T, 3 F, 4 T
Blue) 1 F, 2 T, 3 T, 4 F
Green) 1 F, 2 F, 3 F, 4 T
Yellow) 1 T, 2 F, 3 T, 4 F

Peer Instruction: True or False
A program tries to load a word at X that causes a TLB miss but not a page fault. Which are True or False:
1. A TLB miss means that the page table does not contain a valid mapping for virtual page corresponding to the address X
2. There is no need to look up in the page table because there is no page fault
3. The word that the program is trying to load is present in physical memory.

Red) 1 F, 2 F, 3 F, 4 T
Blue) 1 F, 2 F, 3 T, 4 F
Green) 1 F, 2 T, 3 F, 4 T
Yellow) 1 T, 2 F, 3 T, 4 F

Peer Instruction: True or False
TLBs entries have valid bits and dirty bits. Data caches have them also.
A. The valid bit means the same in both: if valid = 0, it must miss in both TLBs and Caches.
B. The valid bit has different meanings. For caches, it means this entry is valid if the address requested matches the tag. For TLBs, it determines whether there is a page fault (valid=0) or not (valid=1).
C. The dirty bit means the same in both: the data in this block in the TLB or Cache has been changed.
D. The dirty bit has different meanings. For caches, it means the data block has been changed. For TLBs, it means that the page corresponding to this TLB entry has been changed.

Red) 1 F, 2 T, 3 F, 4 T
Blue) 1 F, 2 T, 3 T, 4 F
Green) 1 T, 2 F, 3 F, 4 T
Yellow) 1 T, 2 F, 3 T, 4 F
Agenda

• Virtual Memory Revisted
• Administrivia
• Paging and Performance
• Exceptions, Traps, Interrupts
• Break
• Virtual Machines
• Summary

Administrivia

• Project 3 released, due 8/7 @ midnight
  – Work individually
• No lab Thursday - Work on project, study for final, etc. TAs will be there.
• Final Review - Monday, 8/8.
• Final Exam - Thursday, 8/11, 9am - 12pm 2050 VLSB
  – Part midterm material, part material since.
  – Midterm clobber policy in effect
  – Green sheet provided
  – Two-sided handwritten cheat sheet
    • Use the back side of your midterm cheat sheet!

CS61c In The News

• Software company SofCheck releases “ParaSail” (Parallel Specification and Implementation Language), designed to take a stab at parallel programming.
• “The difference is that it automatically splits a program into thousands of smaller tasks that can then be spread across cores—a trick called pico-threading, which maximizes the number of tasks being carried out in parallel, regardless of the number of cores” -reddit comment
http://www.technologyreview.com/computing/38149
http://groups.google.com/group/parasail-programming-language/browse_thread/thread/7e557034e4f95282

Impact of Paging on AMAT

• Memory Parameters:
  – L1 cache hit = 1 clock cycles, hit 95% of accesses
  – L2 cache hit = 10 clock cycles, hit 60% of L1 misses
  – DRAM = 200 clock cycles (~100 nanoseconds)
  – Disk = 20,000,000 clock cycles (~ 10 milliseconds)
• Average Memory Access Time (HT - Hit Time, MR - Miss Rate)
Impact of Paging on AMAT

- Memory Parameters:
  - L1 cache hit = 1 clock cycles, hit 95% of accesses
  - L2 cache hit = 10 clock cycles, hit 60% of L1 misses
  - DRAM = 200 clock cycles (~100 nanoseconds)
  - Disk = 20,000,000 clock cycles (~10 milliseconds)
- Average Memory Access Time (with no paging):
  - $1 + 5\% \times 10 + 5\% \times 40\% \times 200 = 5.5$ clock cycles
- Average Memory Access Time (with paging) =
  - $5.5 + 5\% \times 40\% \times (1 - \text{MemHitRate}) \times 20,000,000$
  - AMAT if \text{MemHitRate} = 99.9%?
    - $5.5 + 0.02 \times 0.001 \times 20,000,000 = 405.5$
  - AMAT if \text{MemHitRate} = 99.9999%?
    - $5.5 + 0.02 \times 0.000001 \times 20,000,000 = 5.9$

Agenda

- Virtual Memory Revisited
- Administrivia
- Paging and Performance
- Exceptions, Traps, Interrupts
- Break
- Virtual Machines
- Summary

Exceptions and Interrupts

- “Unexpected” events requiring change in flow of control
  - Different ISAs use the terms differently
- Exception
  - Arises within the CPU
    - e.g., Undefined opcode, overflow, syscall, ...
- Interrupt
  - From an external I/O controller
  - Dealing with them without sacrificing performance is difficult

Handling Exceptions

- In MIPS, exceptions managed by a System Control Coprocessor (CP0)
- Save PC of offending (or interrupted) instruction
  - In MIPS: save in special register called Exception Program Counter (EPC)
  - (EPC actually contains PC+4)
- Save indication of the problem
  - In MIPS: saved in special register called Cause register
  - We’ll assume 1-bit
    - 0 for undefined opcode, 1 for overflow
- Jump to exception handler code at address 8000 0180

Handler Actions

- Read Cause register, and transfer to relevant handler
- Determine action required
- If Restartable exception
  - Take corrective action
  - Use EPC to return to program
- Otherwise
  - Terminate program
  - Report error using EPC, cause, ...
**Exceptions in a Pipeline**

- Another kind of control hazard
- Consider overflow on add in EX stage
  - add $1, $2, $1
  - Prevent $1 from being clobbered
  - Complete previous instructions
  - Set *Cause* and *EPC* register values
  - Transfer control to handler
- Similar to mispredicted branch
  - Use much of the same hardware

**Exception Example**

- Exception on `add` in
  - 40 sub $11, $2, $4
  - 44 and $12, $2, $5
  - 48 or $13, $2, $6
  - 4C add $1, $2, $1
  - 50 slt $15, $6, $7
  - 54 lw $16, 50($7)
  - 58 lui $14, 1000
- Handler
  - 80000180 sw $25, 1000($0)
  - 80000184 sw $26, 1004($0)
- Exception Example

  - Time (clock cycles)
  - Instructions: `and`, `or`, `add`, `slt`, `lw`, `lui`
  - Observed: `sw`

**Multiple Exceptions**

- Pipelining overlaps multiple instructions
  - Could have multiple exceptions at once
  - E.g., Page fault in LW same clock cycle as Overflow of following instruction ADD
- Simple approach: deal with exception from *earliest* instruction, e.g., LW exception serviced 1st
  - Flush subsequent instructions
- Called *Precise* exceptions
- In complex pipelines:
  - Multiple instructions issued per cycle
  - Out-of-order completion
  - Maintaining precise exceptions is difficult!

**Imprecise Exceptions**

- Just stop pipeline and save state
  - Including exception cause(s)
- Let the software handler work out
  - Which instruction(s) had exceptions
  - Which to complete or flush
  - May require “manual” completion
- Simplifies hardware, but more complex handler software
- Not feasible for complex multiple-issue out-of-order pipelines to always get exact instruction
- All computers today offer precise exceptions—affects performance though
Beyond Virtual Memory

- Even greater protection than virtual memory
  - E.g., Amazon Web Services allows independent tasks run on same computer
- Can a “small” operating system simulate the hardware of some machine, so that
  - Another operating system can run in that simulated hardware?
  - More than one instance of that operating system run on the same hardware at the same time?
  - More than one different operating system can share the same hardware at the same time?
- Answer: Yes

Solution – Virtual Machine

- A virtual machine provides interface *identical* to underlying bare hardware
  - I.e., all devices, interrupts, memory, page tables, etc.
- Virtualization has some performance impact
  - Feasible with modern high-performance computers
- Examples
  - IBM VM/370 (1970s technology!)
  - VMWare
  - Xen (used by AWS)
  - VirtualBox
  - Microsoft Virtual PC

Virtual Machines

- **Host Operating System:**
  - OS actually running on the hardware
  - Together with *virtualization layer*, it simulates environment for …
- **Guest Operating System:**
  - OS running in the simulated environment
  - Typically, can run many Guest OS’s at once.
  - The resources of the physical computer are shared to create the virtual machines
  - Processor scheduling by OS can create the appearance that each user has own processor
  - Disk partitioned to provide virtual disks

The Unsung Hero of Cloud Computing

- Easy to run an OS or specific configuration independent of hardware.
  - Can support many clients’ demands with a fixed set of hardware.
- Can run multiple OSes / applications per machine
  - Exploit that most applications are idle.
  - Amazon’s EC2 uses VMs to run many system images per physical machine.
**Virtual Machine Monitor**

- Maps virtual resources to physical resources
  - Memory, I/O devices, CPUs
- Guest code runs on native machine in user mode
  - Traps to VMM on privileged instructions and access to protected resources
- Guest OS may be different from host OS
- VMM handles real I/O devices
  - Emulates generic virtual I/O devices for guest

**Example: Timer Virtualization**

- In native machine, on timer interrupt
  - OS suspends current process, handles interrupt, selects and resumes next process
- With Virtual Machine Monitor
  - VMM suspends current VM, handles interrupt, selects and resumes next VM
- If a VM requires timer interrupts
  - VMM emulates a virtual timer
  - Emulates interrupt for VM when physical timer interrupt occurs

**Virtual Machine Instruction Set Support**

- Similar to what need for Virtual Memory
- User and System modes
- Privileged instructions only available in system mode
  - Trap to system if executed in user mode
- All physical resources only accessible using privileged instructions
  - Including page tables, interrupt controls, I/O registers
- Renaissance of virtualization support
  - Current ISAs (e.g., x86) adapting, following IBM’s path

**And in Conclusion, ...**

- Exceptions / Interrupts used to handle unexpected events in processor
  - Sudden change of control => Another control hazard.
- Virtual Machines as even greater level of protection to allow greater level of sharing
  - Enables fine control, allocation, pricing of Cloud Computing